Analysis of the Collision Resistance of RadioGatun using Algebraic Techniques

Charles Bouillaguet, Pierre-Alain Fouque

Analysis of the Collision Resistance of RadioGatún using Algebraic Techniques

Charles Bouillaguet and Pierre-Alain Fouque

Ecole normale supérieure, CNRS, INRIA

Abstract. In this paper, we present some preliminary results on the security of the RadioGatún hash function. RadioGatún has an internal state of 58 words, and is parameterized by the word size, from one to 64 bits. We mostly study the one-bit version of RadioGatún since according to the authors, attacks on this version also affect the reasonably-sized versions. On this toy version, we revisit the claims of the designers and first improve some results. Secondly, given a differential path, we show how to find a message pair colliding more efficiently than the strategy proposed by the authors using algebraic techniques. We experimented this strategy on the one-bit version since we can efficiently find differential path by brute force. Even though the complexity of this collision attack is higher than the general security claim on RadioGatún$\langle 1 \rangle$, it is still less than the birthday paradox on the size of the internal state.

1 Introduction

RadioGatún is a new hash function, proposed in 2006 by Bertoni, Daemen, Peeters and Van Assche at the Second NIST Hash Workshop. This hash function is very interesting to study since its design is not similar to traditional hash functions. It is not a blockcipher-based hash function such as the Davies-Meyer construction of compression function and it does not use the Merkle-Damgård paradigm to transform a compression function into a hash function. This hash function improves a previous design used in the Panama hash function [10]. RadioGatún has an internal state of 58 words; the size of those words, from one to the recommended 64 bits, define the actual size of the internal state.

1.1 Related Work

The Sponge construction. RadioGatún is the current hash function whose design resemble the sponge construction most. This construction differ significantly from the SHA family : the internal building block transform the internal state bijectively, and there is no message expansion: the input blocks are simply injected into the state. The size of the input block is smaller than the internal state, which is also much bigger than the security parameter. The output can be of arbitrarily length.

Sponge functions were introduced in [2, 4], to serve as a reference model for the security of hash functions. Random sponges are an abstraction of a random

function with a finite internal state. In [2], random sponges were shown to be indifferentiable from a random oracle up to a number of query which depends on the "capacity" of the sponge, a part of its internal state.

The Backtracking Attack. The security of the RadioGatún hash function against differential attacks has been initially studied by Bertoni *et al.* in [3]. In fact, the main security analysis has been done on the one-bit version since for this version, differential paths can be found efficiently. If the one-bit version was structurally broken, the the bigger version would be likely to be broken as well. In [3], the authors describe a strategy to find two colliding messages given a differential path and named it the *trail backtracking* attack. This kind of attack improves the statistical attack which tries as many messages as the inverse of the probability of the differential trail. Such an attack has also been mounted on Panama [16, 9] and on Grindahl in [15].

Message Modification and Algebraic Techniques. Expressing the problem of finding a collision as the problem of solving a set of equations is an old technique, that was used to break MD4 first [**?**]. Message modification was used with great success to attack hash functions from the MD and SHA families [**?**,**?**,**?**,**?**]. More advanced algebraic techniques, such as Gröbner bases, were used by [**?**] to improve the message modification part of existing attack against SHA-1.

1.2 Our Results

Our main object of study is the one bit version, RadioGatún$\langle 1 \rangle$. We show that the backtracking attack can be performed more efficiently and without any backtracking. We use Gröbner basis algorithms to compute the set of all states from which colliding messages can be found using a given trail. The main drawback of this attack is that once we have this set, we need other techniques to go from the initial vector to these states. Actually, our method uses statistical trials until one satisfies the equations characterizing the set. The techniques we use heavily rely on the fact that the non-linear function is quadratic and so the differential of such function gives linear conditions on the states.

RadioGatún$\langle 1 \rangle$ has an internal state of 58 bits, and it is conjectured in [3] that differential attacks would cost at least 2^{46}. A first technique using only linear algebra yields collisions in 2^{27} evaluations of the round function. We present a second technique using more sophisticated algebraic tools, most notably Gröbner Basis computations, that produces collisions in less than $2^{24.5}$ evaluations of the round function, for any fixed IV. This is more than one million times faster than what the authors of RadioGatún expected. Both attacks are faster than the birthday paradox on the size of the internal state, but the do not break the security claim of the designers of RadioGatún, since they took a high security margin.

The first attack is trivially applicable to RadioGatún$\langle \ell_w \rangle$ for any value of ℓ_w. The status of the second attack is less clear, but we give some arguments supporting the idea that it will still be applicable when $\ell_w > 1$.

1.3 Organization of the Paper

In section 2, we describe the RadioGatún hash function. Then, we recall the original backtracking attack on RadioGatún presented by its authors and present some surprising experimental results. In section 4, we show how we can improve the backtracking attack using only simple linear algebra. In section 5 and that we can also remove the backtracking by propagating the linear conditions using Gröbner basis computations. Consequently, we derive precise conditions on the states from which colliding pairs can be found.

2 A Brief Description of RadioGatún

RadioGatún is parametrized according to the word length ℓ_w ranging from one to the recommended 64 bits; RadioGatún$\langle \ell_w \rangle$ denote the ℓ_w-bit version of the hash function.

RadioGatún is a hash function based on the sponge paradigm. During the "absorbing" phase, it absorbs an arbitrary number of $3\ell_W$-bit input blocks, and during the "squeezing" phase it produces an arbitrary long output. The input message p is padded, so that its size is a multiple of the input block size. RadioGatún absorbs the input message by alternatively XORing 3 message words into the internal state and applying a bijective round function R until the padded message is entirely read. In each round, $\ell_i = 3 \cdot \ell_W$ bits are absorbed. Note that 6 rounds are required in order that $n/2$ bits are hashed, where n is the size of the internal state. This suggests that colliding message will span over at least 6 rounds. Then, the internal state is mixed using 16 blank iterations of R, and finally the outputs is produced by alternatively extracting 2 words of the internal state and applying the round function until enough bits are extracted. The security of a sponge function is not defined in terms of the digest length (since it can be arbitrarily big), but rather according to another parameter called the *capacity*, which is connected to the size of the internal state. For RadioGatún, the authors made a "flat sponge claim" : More precisely, the authors claim that (truncated) RadioGatún$\langle \ell_W \rangle$ is as strong as a random sponge of capacity $19 \cdot \ell_W$. This mean in particular that it should not be possible to find collisions in less than $2^{9.5 \cdot \ell_W}$, while RadioGatún$\langle \ell_W \rangle$ has an internal state of 58 words ($58 \cdot \ell_W$ bits).

From this description, it is easy to see that a collision into the state at the end of the absorbing phase leads to a collision on the output bits. Consequently, the authors of RadioGatún worry about such collisions and named them *internal collisions*. However, in order to analyze such attacks, it seems that the important parameter is not really the capacity, but rather the half of the size of the internal state. We will see in the following, that if we take this security parameter, we have attacks on the one-bit version of RadioGatún.

In RadioGatún, the state is split into two parts: the *Mill* and the *Belt*. The role of the Belt is to have good long-term diffusion property and uses a simple invertible linear update function, while the goal of the mill is to create confusion

and uses an invertible non-linear update function. The Belt and the Mill interact with each other in each application of the round function.

The Mill a consists of 19 words $a[i]$, and the Belt b is a matrix of 3 rows and 13 columns. An input block x consists of three words $x[i]$. All indices start from 0. We defer the reader to [3] for a complete description of the hash function. Schematically a round of RadioGatún can be described in the following way:

$$b' \leftarrow L_1(b) \oplus L_3(a) \oplus L_2(x)$$
$$a' \leftarrow L_4 \circ \gamma\Big(a \oplus L_5(x)\Big) \oplus L_6(b')$$

where the L_i are bijective linear mappings, and γ is a word-wise bijective quadratic mapping defined by: $\gamma(a)[i] = a[i] \oplus \overline{a[i+1]} \wedge a[i+2]$, where & denotes bitwise AND, and indices are taken modulo 19.

3 The Trail Backtracking attack

3.1 Differential Trails

It seems natural to try a differential attack, considering the successes obtained against the MD and SHA families. We call a differential over the round function a *round differential*; it is a pair (Δ_i, Δ_o). Its *differential probability* (DP) is the proportion of states $\mathbf{s}$ such that $R(\mathbf{s}) \oplus R(\mathbf{s} \oplus \Delta_i) = \Delta_o$. A round differential is *possible* if DP > 0. We may want to take into account not only the internal state, but also the message block entering a round. In this case, a round differential is a triple $(\Delta_i, \Delta_x, \Delta_o)$, and it is satisfied by a state $\mathbf{s}$ and a message block $\mathbf{x}$ if the internal state after the injection of $\mathbf{x}$ satisfies the differential $\big(\Delta_i \oplus F_i(\Delta_x), \Delta_o\big)$. The *(restriction) weight* of a differential is defined by: $W_r(\Delta_i, \Delta_o) = -\log_2 DP(\Delta_i, \Delta_o)$.

Since we will have to track the difference between two parallel hashing processes amongst several iterations of the round function, we are lead naturally to the definition of a *collision trail*; it describes the propagation of the difference on the internal state, when given differences on the input block are applied. Such a trail is a sequence of round differentials: $(\Delta_i^0, \Delta_x^0, \Delta_o^0), (\Delta_i^1, \Delta_x^1, \Delta_o^1), \ldots, (\Delta_i^r, \Delta_x^r, \Delta_o^r)$, where $\Delta_i^0 = 0, \Delta_o^r = 0$, for all $1 \leq k \leq r, \Delta_{k-1}^o = \Delta_k^i$. For each round k, the trail enforces that if the internal states satisfy $\mathbf{s^k} \oplus \mathbf{s'}^k = \Delta_i^k$, and the input message blocks satisfy $\mathbf{x}^k \oplus \mathbf{x'}^k = \Delta_x^k$, then after R the output state pair has difference Δ_o^k. If one finds an input state $\mathbf{s^0}$ and a sequence of message blocks $\mathbf{x^0}, \ldots, \mathbf{x^r}$ satisfying all the conditions imposed by the trail, then one has found a collision. The probability that a random message follows the trail is the differential probability (DP) of the trail, and the *differential weight* of $\mathcal{T}$ is defined by $W_r(\mathcal{T}) = -\log_2 DP(\mathcal{T})$.

3.2 The Trail Backtracking Attack

Given a r-round differential trail and an initial state, a naive way to look for a collision would be to try random sequences of r message blocks satisfying the

differences specified by the trail until a collision is found. The expected workload of this attack is $r/DP(\mathcal{T})$ evaluations of R. It may very well happen that the input message passes some of the first rounds with the right difference, but then diverges from the trail. This message has an interesting *prefix*, but in the naive attack it is simply thrown away. Additionally, it is useless to hash the end of the message, since we could know in the middle that it would not follow the trail.

In the backtracking attack, however, a right prefix is reused as much as possible. It can be seen as an analogous of Wang's message modification on Davies-Meyer-type compression functions, but adapted to the alternating-input framework. Suppose we have a message that passes the first k rounds, but not the $(k+1)$-th. Either the choice of $\mathbf{x^k}$ was bad and by choosing another block we can pass the $(k+1)$-th round, or the previous choices of $\mathbf{x^0}, \ldots, \mathbf{x^{k-1}}$ were bad, and we have to reconsider them (this is what we will call "backtracking").

More precisely, if a right pair enters round k, the difference at the input of the round function will be the same regardless of the value of the input block $\mathbf{x^k}$, as long as it satisfies the specified difference Δ_x^k. Therefore, this right pair can be turned into 2^{ℓ_i} right pairs by simply enumerating all possible values of $\mathbf{x}$. If this results in a right outgoing pair, we can proceed to the next round, and otherwise, we have to backtrack to the previous round. This can be seen as the depth-first exploration of a big 2^{ℓ_i}-ary tree in which nodes are labeled with internal state values and edges are labeled with message blocks (the root being labeled by $\mathbf{s^0}$).

BT_Attack(s, k) :
Given a right pair entering the k-th round, try to go further along a given trail $\mathcal{T}$ or backtrack.

- If $k = |\mathcal{T}|$, then a collision has been found
- For all possible input block $\mathbf{x^k}$ do
  - if the state $\mathbf{s}$ along with the input block $\mathbf{x}$ pass the k-th round differential of $\mathcal{T}$, *i.e.* if

$$R(\mathbf{s} \oplus F_i(\mathbf{x})) \oplus R\left(\left(\mathbf{s} \oplus \Delta_i^k\right) \oplus F_i\left(\mathbf{x} \oplus \Delta_x^k\right)\right) = \Delta_o^k$$

  then invoke **BT_Attack(**$R(\mathbf{s} \oplus F_i(\mathbf{x})), k+1$)

Fig. 1. Pseudocode of the trail backtracking attack.

It may very well happen that the input state $\mathbf{s^0}$, which can be chosen at random by hashing a random message for example, cannot possibly lead to a collision along $\mathcal{T}$. In that case, we just have to generate a new one.

3.3 The Original Complexity Analysis of [3]

The authors of [3] give a generic complexity analysis of the trail backtracking attack. They always assume that the conditions imposed by the round differentials

are independent from each other, which means that:

$$W_r(\mathcal{T}) = \sum_{k=0}^{r-1} W_r(\Delta_i^k, \Delta_x^k, \Delta_o^k)$$

Following [3], we assume that we will try N pairs of (random) input state $\mathbf{s^0}$ before finding a collision. We count the number of right pairs entering and going out of each round ; the round with the most incoming pairs is called the *crowded round*, and the round with the less outgoing pairs is called the *lonesome round*. If q pairs enter round k, then we can expect $q \cdot 2^{\ell_i - W_r(\Delta_i^k, \Delta_x^k, \Delta_o^k)}$ pairs to go out. We therefore define the *excess weight* in round k to be:

$$W_e(k) = \sum_{j=0}^{k-1} \big(W_r(\Delta_i^j, \Delta_x^j, \Delta_o^j) - \ell_i\big)$$

The total expected number of pairs entering round k is $N \cdot 2^{-W_e(k)}$, and the expected number of pairs going out of round k is $N \cdot 2^{-\ell_i - W_e(k+1)}$. The analysis now proceeds in two steps:

Evaluate N We assume that the attack succeeds as long as at least one pair goes out of each round. This imposes $N \geq 2^{\ell_i + W_e(k+1)}$ for all $1 \leq k < r$. This condition is satisfied by setting $N = 2^{\ell_i + \max_k W_e(k+1)}$.

Evaluate the Workload According to [3], the workload can be approximated by the number of pairs entering the crowded round : $L(\mathcal{T}) \simeq \max_k N \cdot 2^{-W_e(k)} = N \cdot 2^{-\min_k W_e(k)}$. Therefore, by using the previous result, we define the *backtracking cost*:

$$C_b(\mathcal{T}) = \ell_i + \max_{0 \leq j < k \leq r} W_e(k) - W_e(j)$$

The workload of the trail backtracking attack is then: $L(\mathcal{T}) \simeq 2^{C_b(\mathcal{T})}$.

The authors of [3] present arguments that RadioGatún resists the trail backtracking attack, using this complexity analysis. In particular, on RadioGatún⟨1⟩, where the internal state is 58-bit long, they performed an extensive search and did not find collision trail with backtracking cost smaller than 46. If there were no better trail, this would imply that the trail backtracking attack could not possibly be faster than exhaustive search on the one-bit version. Because the description of RadioGatún makes use of intra-word rotation, an operation that has no effect on the one-bit version, it is likely that the many-bit version have better diffusion, and therefore are stronger.

We emphasize that the differential weight is not a relevant indicator, because the backtracking attack may dramatically reduce the cost of finding a collision. A similar phenomenon occurs in the backtracking attack against Grindalh, or in the differential attacks based on message-modification against MD5.

3.4 Experimentation with the Backtracking Attack

We implemented the two required steps of the trail backtracking attack to find collisions : finding differential trails, and actually finding colliding messages using a given trail. Note that [3] only present experimental results about the former. The C++ programs that we developed are available on the webpage of the first author.

Finding Differential Trails. In RadioGatún$\langle 1 \rangle$, it is possible to find collisions by brute force, and a collision describes a collision trail. We therefore looked for collisions extensively, and collected the corresponding trails.

The authors of [3] communicated to us their best collision trail on RadioGatún$\langle 1 \rangle$, that we will note $\mathcal{T}_1$. It is completely defined by two colliding messages sharing a 7-block prefix followed by a 8-block colliding part. In octal notations, the two messages are : 0364220 64172767 and 0364220 20435061. The differential weight of this trail is 63, and its backtracking cost is 46.

We eventually found a 7-round trail (called $\mathcal{T}_2$) with backtracking cost 31 and differential weight 45. This surprising result was obtained while looking for 7-round trails, by initializing the internal state with a 9-block random prefix. $\mathcal{T}_2$ is defined by the following colliding messages, again in octal notation: 476356301 6336565 and 476356301 4250471. With the trail backtracking attack, this gives a collision in an expected 2^{31} effort, which is still above the birthday bound.

Searching for a Collision. We used these two trails to find collisions on RadioGatún$\langle 1 \rangle$. It may be argued that we needed to find collisions (to get the trails) before actually being able to find collisions, but there may very well be other methods of finding trails, and we did not consider this problem. Moreover, once a good collision trail is found, it can be used to find collision from any value of the internal state. At the very least, if a technique were found to efficiently find chosen-IV collisions, it could generate collision trails, and therefore be used as a preprocessing step in our attack.

On average, the trail backtracking attack succeeds with 2^{29} evaluations of the round function (when using $\mathcal{T}_2$), which is exactly the complexity of the birthday bound. With $\mathcal{T}_1$, the attack succeeds in $2^{34.5}$ (in average), when the announced complexity was 2^{46}.

In order to get some insight to why the collision search procedure succeeds faster than expected, we observed the number of right pairs going in and out of each round. Figure 2 shows these numbers when the first collision is found (using $\mathcal{T}_1$). The results are similar if we average them on 100 collisions, or to what is obtained with other trails. After round 2 or 3, the pairs pass the next round with abnormally high probability. Apparently, the round differentials are not independent. This would explain why the first rounds have a tendency to "filter" good pairs that pass the subsequent rounds more easily. This may be very specific to the one-bit version of RadioGatún, though. It may also be specific to the way the collision trails were obtained (by actually computing a collision).

round	in. pairs	out. pairs	$W_r(\Delta_i^k, \Delta_x^k, \Delta_o^k)$	experimental weight
0	32.01	30.01	2	2.0
1	32.82	25.82	7	7.0
2	28.62	16.66	11	12.0
3	19.46	12.88	11	6.6
4	15.69	7.61	10	8.1
5	10.42	6.02	11	4.4
6	8.82	0.00	11	8.8
7	0.00	0.00	0	0.0

Fig. 2. (log_2 of the) Number of pairs going in and out of each round in $\mathcal{T}_1$; comparison with the weight of each round.

In itself, the trail backtracking attack does not break RadioGatún.

4 Improving the Backtracking Attack on RadioGatún$\langle \ell_W \rangle$

In the next sections, we focus on improving the efficiency of the trail backtracking attack whose complexity is above the birthday bound, using algebraic techniques, as suggested by the authors of RadioGatún themselves.

The only non-linear part in RadioGatún is the "mill function", and more specifically its first component γ. The specific properties of γ are extensively studied in [8, chapter 6]. For now, let us notice that γ has algebraic degree 2 over $\mathbb{F}_2$ (each bit of $\gamma(a)$ can be expressed as a quadratic form in the input bits). Because the rest of the round function is linear, the whole round function R can be expressed as a tuple of $58 \cdot \ell_W$ polynomials of degree 2 in $58 \cdot \ell_W$ variables over $\mathbb{F}_2$ (we denote by $\mathbb{F}_2[\mathbf{s}]$ the set of all polynomials over $58 \cdot \ell_W$ variables corresponding to the bits of the internal state).

It is well-known that if a function is quadratic, then its differential is *linear*. This is the key idea in this preliminary algebraic analysis of RadioGatún. Let us consider the set of internal states $\hat{\mathbf{s}}$ after input injection satisfying the round differential (Δ_i, Δ_o). These state satisfy the following equation: $R(\hat{\mathbf{s}}) \oplus R(\hat{\mathbf{s}} \oplus \Delta_i) = \Delta_o$. Even though R is quadratic, this equation is only *linear* in $\hat{\mathbf{s}}$. Therefore, we know that all the values of $\hat{\mathbf{s}}$ satisfying it lie in an affine space, and thus can be characterized by linear conditions on $\hat{\mathbf{s}}$. These conditions depend on Δ_i and Δ_o, and can be computed efficiently using linear algebra. We denote by $\mathcal{C}(\Delta_i, \Delta_o)$ (or $\mathcal{C}^k$) these conditions. The state entering the round function is given by $\hat{\mathbf{s}} = \mathbf{s} \oplus F_i(\mathbf{x})$. Therefore, conditions on $\hat{\mathbf{s}}$ give two kinds of information:

1. linear conditions on the bits of $\mathbf{s}$.
2. linear conditions between bits of $\mathbf{s}$ and bits of $\mathbf{x}$.

The former can be used to detect incoming pair that will never give rise to an outgoing pair, for any value of $\mathbf{x}$. This allows to stop the exploration of dead branches of the tree earlier. The latter directly gives us some bits of $\mathbf{x}$,

k	$\mathbf{x}^k[0]$	$\mathbf{x}^k[1]$	$\mathbf{x}^k[2]$
0	$\mathbf{a}[16]$		$\mathbf{a}[18]+1$
1	$\mathbf{a}[16]+1$		
2			
3		$\mathbf{a}[17]+\mathbf{a}[16]+\mathbf{x}[0]$	$\mathbf{a}[18]$
4	$\mathbf{a}[15]+\mathbf{a}[16]+1$	$\mathbf{a}[17]+1$	$\mathbf{a}[18]$
5		$\mathbf{a}[17]+\mathbf{a}[16]+\mathbf{x}[0]+1$	$\mathbf{a}[18]$
6			

Fig. 3. When using $\mathcal{T}_2$, some bits of the input blocks $\mathbf{x}^k$ are determined by some bits of the incoming state. An empty cell means that the corresponding bit has to be chosen by the attacker.

as linear combinations of bits of $\mathbf{s}$, and thus allow us to filter the values of $\mathbf{x}$ that do not yield a right outgoing pair. Using these conditions, we can decrease the amount of useless trials in the backtracking attack. Figure 3 shows which bits of the input message are determined by the internal state, for the trail $\mathcal{T}_2$. The complete set of conditions is given in fig. 4.1. It must be noted that these conditions can be computed efficiently for all values of the word size ℓ_W (the linear algebra involved is cubic in the word size).

4.1 Experimental Results

We implemented the improved backtracking attack on RadioGatún$\langle 1 \rangle$, using the same two trails $\mathcal{T}_1$ and $\mathcal{T}_2$, so that the result can be compared with the regular attack. For $\mathcal{T}_2$, collisions are found on an average of 2^{27} evaluations of R, which means a speedup of 4 compared to the regular attack. Note that this is below the birthday bound. Annex 4.1 shows the local conditions imposed by $\mathcal{T}_2$. For $\mathcal{T}_1$, 2^{32}, which is more than 5.5 times faster than the original attack.

5 The Backtrackingless Backtracking attack

While the technique described in the previous section reduces the amount of backtracking by allowing an earlier filtering of pairs that will not lead to a collision, it does not prevent all backtracking. The reason for this is that this filtering is only *local*: at round k, we cannot yet filter pairs that will not pass round $k+1$. In this section, we address this issue. We show that it is possible to avoid *all* backtracking by propagating equations backwards from the last round to the first round of the trail. We get a set of equations on the internal states entering the first round ; if a state $\mathbf{s^0}$ satisfies these equations, then we can generate a few collisions at a negligible cost. We achieve some kind of *global* filtering, because we filter at the first round all the pairs that will not pass any

k	$\mathcal{C}\left(\Delta_i^k, \Delta_x^k, \Delta_o^k\right)$	
0	$x[0] + a[16]$	$x[2] + a[18] + 1$
1	$x[0] + a[16] + 1$	$a[0]$
	$a[2]$	$a[3] + 1$
	$a[5] + a[4]$	$a[6]$
	$a[8]$	$a[10] + 1$
	$a[12]$	$a[14] + 1$
2	$a[4] + 1$	$a[7] + a[5] + 1$
	$a[7] + a[6] + 1$	$a[8] + 1$
	$a[13]$	$a[15]$
3	$x[0] + a[17] + a[16] + x[1]$	$x[2] + a[18]$
	$a[1] + 1$	$a[3]$
	$a[13] + 1$	$a[15]$
4	$a[15] + x[0] + a[16] + 1$	$a[17] + x[1] + 1$
	$x[2] + a[18]$	$a[3] + a[0] + 1$
	$a[3] + a[1]$	$a[3] + a[2]$
	$a[4]$	$a[6]$
	$a[7] + a[8] + 1$	$a[9]$
	$a[14] + 1$	
5	$x[0] + a[17] + a[16] + x[1] + 1$	$x[2] + a[18]$
	$a[0]$	$a[2] + 1$
	$a[4]$	$a[7]$
	$a[8] + a[10]$	$a[9] + a[10] + 1$
	$a[11] + 1$	$a[15] + 1$

Fig. 4. Conditions imposed at the beginning of each round by $\mathcal{T}_2$

of the subsequent rounds (we "push" all the conditions at the root of the tree). We propose to name this attack the *Backtrackingless Backtracking attack*[1].

In the previous section, we showed how to generate a set of conditions $\mathcal{C}^k$ such that if a state $\mathbf{s}$ satisfies these conditions, then the pair $\left(\mathbf{s}, \mathbf{s} \oplus \Delta_i^k\right)$ will pass round k. In order to pass round $k+1$, the states going out from round k must also satisfy $\mathcal{C}^{k+1}$. Our objective is to express a new set of conditions on $\mathbf{s}$ such that if these conditions are satisfied, then $\mathbf{s}$ satisfies $\mathcal{C}^k$ *and* $\mathbf{s}'$ satisfies $\mathcal{C}^{k+1}$. We achieve our objective of propagating all the conditions backwards to the first round by recursively applying this process.

5.1 Description

We use standard notions and notations for commutative algebra, that can be found for example in [7]. Formally, we say that a (polynomial) condition (or constraint, or equation) on $\mathbf{s}$ is a polynomial of the ring $\mathbb{F}_2[\mathbf{s}]$ (*i.e.*, a polynomial in which the variables are bits of $\mathbf{s}$). A condition P is satisfied by $\mathbf{s}$ if P vanishes

[1] Its name is reminiscent of the *inductionless induction* of [6] or of the *splittingless splitting* of [13].

when the variables are substituted with the actual values of bits in $\mathbf{s}$. We can then write $P(\mathbf{s}) = 0$, or, using the notation from the area of logics, $\mathbf{s} \models P$. The set of states satisfying P is then the set of zeroes of P. We will also have to consider the conjunction $\mathcal{C}$ of several such conditions (*i.e.*, systems of polynomial equations). A convenient way to represent such a system is to consider the polynomial *ideal* I generated by the polynomials in $\mathcal{C}$. It contains all the polynomial combinations of its generators, that is, all the polynomial "consequences" of the original equations. The set of states satisfying $\mathcal{C}$ is the set of all common zeroes of all the polynomials in I, which is called the *affine variety* $\mathbf{V}(I)$ associated to I. We say that a set of conditions $\mathcal{D}$ is a consequence of another system $\mathcal{C}$ if $I_{\mathcal{D}} \subseteq I_{\mathcal{C}}$. We note $\mathcal{C} \Rightarrow \mathcal{D}$ to describe this situation.

When expressing conditions about the output of the round function, we introduce $58 \cdot \ell_W$ more variables $\mathbf{s}'$, corresponding to the bits of the output. The equations of R are actually equations in $\mathbb{F}_2[\mathbf{s}, \mathbf{s}']$. We will note I_R the ideal of $\mathbb{F}_2[\mathbf{s}, \mathbf{s}']$ generated by the equations of R; its affine variety contains all the tuples $(\mathbf{s}, \mathbf{s}')$ such that $\mathbf{s}' = R(\mathbf{s})$. From a geometric point of view, these equations describe the *graph* of the function R (in the same fashion that $y - x^2 = 0$ describes a parabola). Later on, we will use a different representation of these equations, that still describe the same graph.

We need a last tool before defining formally the objects we wish to compute. We need to express conditions on the input of the $(j+1)$-th round as conditions on the output of the j-th round. This is simply done by renaming variables. We define the renaming function $\rho : \mathbb{F}_2[\mathbf{s}] \to \mathbb{F}_2[\mathbf{s}']$ as $\rho(\mathbf{s}_j) = \mathbf{s}'_j$. This renaming can be extended to operate on ideals : $\rho(I) = \left\{\rho(P) \mid P \in I\right\}$. It is straightforward to check that $\rho(I)$ is still an ideal.

New Sets of Conditions. Given a r-round trail $\mathcal{T}$, a sequence of r input blocks $\left(\mathbf{x}^k\right)_{0 \leq k < r}$ and an internal state $\mathbf{s}^0$, we note $\mathbf{s}^{i+1} = R\left(\mathbf{s}^i \oplus F_i\left(\mathbf{x}^i\right)\right)$. Our objective is to build r sets of conditions $\mathcal{D}^k, 0 \leq k < r$ such that if $\mathbf{x}^k \models \mathcal{D}^k$, then for all $j \geq k$, $\mathbf{x}^j \models \mathcal{C}^j$ (if the internal state at the input of round k satisfies the conditions $\mathcal{D}^k$, then we know for sure that it will lead to a collision because it satisfies all the subsequent sufficient conditions $\mathcal{C}^j$, for $k \leq j$). In particular, if we are able to find an internal state satisfying $\mathcal{D}^0$, then we get a collision nearly for free. Intuitively, our objective is to transfer simultaneously all the conditions $\mathcal{C}^i$ at the beginning of each round to conditions on the internal state $\mathbf{s}^0$ entering the first round. It must be noted that the authors of [3] mentioned the possibility to propagate conditions on the input of the lonesome round to the input of the preceding rounds. Here, we propagate conditions on the internal state.

From the definition of $\mathcal{D}^k$, we can first deduce that $\mathcal{D}^k \Rightarrow \mathcal{C}^k$, and then that if $\mathbf{s}^k \models \mathcal{D}^k$, then $\mathbf{s}^{k+1} \models \mathcal{D}^{k+1}$. We also know that there are no conditions on the output of round r (because a collision is already obtained), and therefore: $\mathcal{D}^{r-1} = \mathcal{C}^{r-1}$. For $0 \leq j < r-1$, we can now define $\mathcal{D}^j$ by:

$$\mathcal{D}^j = \left(\mathcal{C}^j + I_R + \rho\left(\mathcal{D}^{j+1}\right)\right) \bigcap \mathbb{F}_2[\mathbf{s}]$$

Informally, $\mathcal{D}^j$ is the ideal obtained by writing together the constraints $\mathcal{D}^{j+1}$ on $\mathbf{s}'$, the equations of the round function and the constraints $\mathcal{C}^j$ on $\mathbf{s}$. By taking its intersection with $\mathbb{F}_2[\mathbf{s}]$, we eliminate all the polynomials containing a variable from $\mathbf{s}'$. This amounts to considering the consequences of these equations that can be expressed using only the variables of $\mathbf{s}$ – a process known as *eliminating* the variables $\mathbf{s}'$.

Computing the $\mathcal{D}^j$'s. The Hilbert Basis theorem tells us that, like all ideals of a polynomial ring, $\mathcal{D}^j$ admits a finite number of generators; moreover, they can be computed using a computer algebra system: compute a Gröbner basis G of $\mathcal{C}^j + I_R + \rho(\mathcal{D}^{j+1})$ for the lexicographic ordering (or a suitable elimination ordering). The basis G generates $\mathcal{C}^j + I_R + \rho(\mathcal{D}^{j+1})$, but the *elimination theorem* (see [7]) additionally tells us that $G \cap \mathbb{F}_2[\mathbf{s}]$ generates $\mathcal{D}^j$. Now, we claim that $\mathbf{V}\left(\mathcal{D}^j\right)$ is exactly the set of all the states that will pass the end of the trail. This is in fact a consequence of the *extension theorem*: if $\mathbf{s} \in \mathbf{V}\left(\mathcal{D}^j\right)$, then there exists an "extension" value $\mathbf{s}'$ such that $(\mathbf{s}, \mathbf{s}') \in \mathbf{V}\left(\mathcal{C}^j + I_R + \rho\left(\mathcal{D}^{j+1}\right)\right)$. Because we included the equations of R, this value is necessarily $R(\mathbf{s})$. The conclusion follows by induction on the number of rounds.

To complete the attack, we need to find a message yielding an internal state $\mathbf{s}$ satisfying $\mathcal{D}^0$, starting from the IV (which is the null state); then we would automatically get a collision without any backtracking. Note that being able to just determine a "standalone" state in $\mathbf{V}\left(\mathcal{D}^0\right)$ would give a chosen-IV collision attack.

Finding points in an affine variety is difficult in the general case, but becomes easier when a Gröbner basis of the corresponding ideal is known (and it is *very* easy when a Gröbner basis is known for the *lexicographic* ordering). Here, as it result from the process of elimination, $\mathcal{D}^0$ form a Gröbner basis for a certain ordering, which depends on the ordering used for the elimination process. It could be chosen so that $\mathcal{D}^0$ form a lexicographic Gröbner basis (using a block order where the non-eliminated variables are ordered lexicographically), but this may make the elimination process slower. In any case, order change algorithms could be used, such as the Gröbner Walk [5] or FGLM [12].

Reaching $\mathcal{D}^0$. To find real collisions, we need to be able to reach $\mathbf{V}\left(\mathcal{D}^0\right)$ starting from the null state. The problem of finding a collision thus reduces to the problem of reaching a state satisfying a set of polynomial conditions. This formulation of the problem is again reminiscent of message modification techniques. This suggest that such powerful techniques could be used here. We did not investigate this problem in detail, and we only tried to hash random messages until all the conditions are satisfied. In this case, the complexity of finding a collision is related to the cardinality of $\mathbf{V}(\mathcal{D}^j)$.

The representation of the condition set $\mathcal{D}^0$ on the initial conditions is not unique. In our case, it forms a Gröbner basis, which is certainly interesting. We have some freedom in the choice of the ordering. The Graded Reverse Lexicographic order produces the system of lowest possible degree, but usually returns

a system with more equations than when using the lexicographic ordering (which yields equations of higher degree).

5.2 Implementation and Experimental Results

We implemented the backtrackingless backtracking attack, using an off-the-shelf computer algebra system to perform the algebraic computations, and then we adapted our collision-finding program to use these conditions.

Propagating Conditions. Back to our two trails $\mathcal{T}_1$ and $\mathcal{T}_2$, the process of computing the conditions $\mathcal{D}_0$ involves nontrivial algebraic computation. We used the implementation of the F4 [11] algorithm in the MAGMA computer algebra system to obtain the Gröbner bases. We expected these computations to be very hard (the systems have 100+ variables, and contains the equations of R). It is usually not possible to compute a Gröbner basis of the equations describing directly a cryptographic primitive – MAGMA ate 8Gb of memory and crashed when we tried to compute a Gröbner basis of I_R. However, the computations of our sets of conditions were not only possible, but also unexpectedly fast (less than a second). Computing $\mathcal{D}^0$ for a given trail is usually a matter of less than five seconds on a desktop computer. Even more surprisingly, the conditions $\mathcal{D}^j$ are almost always *linear*, for all trails, except when $j = 0$ on some trails.

- For $\mathcal{T}_2$ in particular, the conditions $\mathcal{D}^5$, $\mathcal{D}^4$, $\mathcal{D}^3$ and $\mathcal{D}^1$ are linear. $\mathcal{D}^2$ contains a few equations of degree 2, and $\mathcal{D}^0$ contains one equation of degree 3, along with 97 quadratic and 15 linear equations.
- For $\mathcal{T}_1$, only $\mathcal{D}^0$ is non-linear ; it contains 26 quadratic and 26 linear equations.

This means that the size of conditions propagated through the round function does not blow up exponentially with the number of round passed. This was unexpected, because the size of the equations describing $R^{(k)}$ grows exponentially with k. In fact, the local conditions computed in section 4 play a *crucial* role here: the Gröbner basis computation are *much* faster and more tractable if there are a few linear conditions on the internal state entering the round. In particular, $\mathcal{C}^0$ is usually almost empty for many trails, and the conditions $\mathcal{D}^0$ are usually bigger and of higher degree than the others.

Actually Finding Collisions. Our collision-finding program just hashes random messages and checks if the resulting internal state satisfies $\mathcal{D}_0$. If it is the case, the previous version of the backtracking attack is run, and succeeds without backtracking. The performance of our straightforward implementation is the following: for $\mathcal{T}_2$, a collision is found with about $2^{24.5}$ evaluations of the round function ($2^{29.5}$ for $\mathcal{T}_1$, which means a speedup of 32 compared to the original attack).

In addition, we estimated the size of the set of states leading to a collision by Monte-Carlo sampling : the probability that a random message yields a state

from which a collision is possible for $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) is $2^{-28.42}$ (resp. $2^{-23.4}$). This means that for $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) we have $|\mathcal{T}_1| \simeq 2^{29.6}$ (resp. $2^{34.6}$). It is worth noting that the trail that has the best backtracking cost still yields the biggest affine variety. We also note that the running time of our simple implementation is relatively well-correlated to the cardinalities of the affine varieties.

5.3 About the Structure of the Equations

In this short section, we give a few elements in order to explain why the algebraic attack is successful. We discuss the case of the one-bit version, but the discussion also apply to the bigger versions, as we may notice. There are 19 mill equations that we denote by $f_1, \ldots, f_{19}$.

Peeling Off the Diffusion Layer. Compared to algebraic attacks on block ciphers, the situation is quite easier here. First of all, we are not facing a monolithic cipher where all the internal state is unknown but we can attack each round independently of the others (the conditions $\mathcal{S}^j$ bridge the gap between the isolated rounds). The equations we are manipulating therefore only represent a single round, which is much weaker than the whole construction.

Second, the hardness of solving the equations associated with a block cipher come from the alternation of a simple non-linear part (the S-Boxes, or the γ function here) with a linear diffusion layer. Since we are considering a single round here, it is possible to "peel off" the diffusion layer, and to expose the non-linear core directly, by considering a linear combination of the original equations:

$$\begin{pmatrix} g_1 \\ \vdots \\ g_{19} \end{pmatrix} = L_4^{-1} \times \begin{pmatrix} f_1 \\ \vdots \\ f_{19} \end{pmatrix} = \boldsymbol{\gamma}\Big(a \oplus L_5(x)\Big) \oplus \text{linear terms}$$

Thus, it is relatively equivalent to perform our analysis on the equations of γ and on the equations of the mill function. Recall that $\gamma(a)[i]$ is given by:

$$\gamma(a)[i] = a[i+1]a[i+2] \oplus a[i+2] \oplus a[i] \oplus 1$$

Sparsity of the Equations. Computing a Gröbner base of the equations of γ is not easy (MAGMA takes about 10 minutes on a fast machine and requires 2.8 Gbytes of memory to do so, for the degree reverse lexicographic order). However, these equations have a specific structure that can be exploited. They are extremely sparse, each containing only one quadratic term. Moreover, each variable appear in exactly two quadratic terms. This means that if the value of a variable is fixed, two equations become linear. To illustrate how bad a property this is, let us consider a random quadratic form in n variables. It is shown in [14] that on fields of characteristic 2, any quadratic form becomes a special standard form $f = \sum_{i=0}^{n/2} x_{2i}x_{2i+1}$ under the right change of variables

(some details omitted for the sake of simplicity). This means that we may have to fix about $n/2$ variables in the new basis before the form becomes linear.

Let us go back to our main computational problem, namely the computation of a Gröbner basis of the ideal generated by $\mathcal{C}^j + I_R + \rho\left(\mathcal{D}^{j+1}\right)$, for a suitable elimination ordering. Along with the equations of R are the linear conditions $\mathcal{C}^j$ imposed on the input bits of each round. These conditions, shown in fig. 4.1, often fix the value of one bit. Therefore, in conjunction with the removal of the diffusion layer, they can be used to *dramatically* simplify the equations ■< .mine of I_R. This explain not only why the algebraic computations are fast, but also why the propagated conditions are mostly linear. In fact, computer algebra systems are able to perform these simplifications *automaticaly.* ======= of I_R. In fact, these simplifications are able performed *automatically* by most computer algebra systems. This explains why the elimination process results in mostly linear equations, and terminates so fast. ■> .r721

The REDUCTION Algorithm. Let $\mathcal{B}$ be a set of polynomials. A polynomial P is said to be reduced for $\mathcal{B}$ if no monomial of P lies in the ideal generated by the head terms of $\mathcal{B} - P$. Intuitively, this means that P cannot be "simplified" by a polynomial combination of elements in $\mathcal{B}$. A Basis $\mathcal{B}$ is said to be reduced if each $P \in \mathcal{B}$ is reduced for $\mathcal{B} - P$ (the polynomials of $\mathcal{B}$ cannot simplify each other). The REDUCTION algorithm, which gives a reduced basis from an arbitrary basis, is described in [7, chapter 2, paragraph 7], and in [1, figure 5.2]. Note than when applied to linear polynomials only, it is actually (a version of) the Gaussian reduction algorithm.

REDUCTION is often invoked automatically in computer algebra systems before and after the computation of a Gröbner basis. When the graded-reverse lexicographic ordering is used, it removes some of the quadratic terms in the equations of γ by substituting the linear equations of $\mathcal{C}^j$ in them. Additional tuning of variable order does not seem to be necessary to obtain satisfactory results. However, ordering the variables in the following way: $\mathbf{a}' < \mathbf{a} < \mathbf{x}$ peels off the diffusion layer very nicely, by keeping the number of quadratic term close to the minimum, and making the 19 quadratic terms of γ the head terms of the 19 equations.

6 Extension to $\ell_w > 1$

The main interest in studying RadioGatún$\langle 1 \rangle$, according to [3], is that a collision trail for the one-bit version could be transformed into a collision trail for any n-bit version, with an increased differential weight. In this section, we briefly survey how the result presented in this paper apply to the case where $\ell_W > 1$.

The backtracking attack with local filtering presented in section 4 can be mounted for any value of ℓ_W without any difficulty, as its complexity is polynomial in ℓ_w.

The backtrackingless backtracking attack may be more difficult to implement, as we have no upper-bound on the complexity of the Gröbner basis computations

involved in the attack. However, all the arguments given in section 5.3 still apply to the multi-bit case ; the diffusion layer can be gotten rid of as efficiently as in the one-bit case. Then, the multi-bit version of γ is actually a collection of ℓ_w copies of the one-bit version of γ operating independently (the diffusion layer is supposed to connect them).

Unfortunately, we did not implement the attack in the multi-bit case, because we were not able to find any possible differential trail for any value of $\ell_w > 1$. The heuristic argument of [3] regarding the extension of trails from 1-bit to n-bit assumes that the conditions imposed by the round differentials are independent. As we have seen earlier, this is not the case. All the possible trails we knew for the 1-bit version turned out to be impossible to extend to n-bit versions (the round differentials seem to impose contradictory conditions).

In any case, we believe that our technique may come in handy when collision trail will be found for RadioGatún$\langle \ell_w \rangle$ with $\ell_w > 1$ though.

7 Conclusion

We presented an improvement to the trail backtracking attack introduced by the authors of RadioGatún, and which is reminiscent of the well-known message modification applied against the MD and SHA family. We are able to give an algebraic characterization of the internal states that can lead to a collision along a given trail. Finding a message mapping the IV to a state satisfying all these conditions remains an open problem, which is also reminiscent of message modification.

These preliminary remarks on RadioGatún invite some comments : the fact that the round function is only quadratic seems to be exploitable in unpredictable ways. It would be safe to consider functions of higher degree, but the hashing speed would probably be affected. Alternatively, increasing the diffusion effect of the belt in order to exploit the non-linearity of the mill function further seems to be a potential solution to make the backtracking cost of collision trails higher.

Acknowledgement We thank Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche for many useful discussion and comments. The authors are indebted to Christophe de Cannière who helped us with the trail-finding program.

References

1. Becker, T., Weispfenning, V., Kredel, H.: Gröbner bases: a computational approach to commutative algebra. Springer-Verlag, London, UK (1993)
2. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indifferentiability of the sponge construction. In Smart, N.P., ed.: EUROCRYPT. Volume 4965 of Lecture Notes in Computer Science., Springer (2008) 181–197

3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: RadioGatún, a Belt-and-Mill Hash Function. Presented at Second Cryptographic Hash Function Workshop, August 24-25, 2006, Santa Barbara, California (August 2006) `http://radiogatun.noekeon.org/`.
4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge functions. Presented at ECrypt Hash Function Workshop, May 24, 2007, Barcelona, Spain (May 2007)
5. Collart, S., Kalkbrener, M., Mall, D.: Converting bases with the gröbner walk. J. Symb. Comput. **24**(3/4) (1997) 465–469
6. Comon, H.: Inductionless induction. In Robinson, J.A., Voronkov, A., eds.: Handbook of Automated Reasoning. Elsevier and MIT Press (2001) 913–962
7. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra (Undergraduate Texts in Mathematics). Springer (February 2007)
8. Daemen, J.: Cipher and hash function design. Strategies based on linear and differential cryptanalysis. PhD thesis, Katholieke Universiteit Leuven (March 1995)
9. Daemen, J., Assche, G.V.: Producing collisions for panama, instantaneously. In Biryukov, A., ed.: FSE. Volume 4593 of Lecture Notes in Computer Science., Springer (2007) 1–18
10. Daemen, J., Clapp, C.S.K.: Fast hashing and stream encryption with panama. In Vaudenay, S., ed.: FSE. Volume 1372 of Lecture Notes in Computer Science., Springer (1998) 60–74
11. Faugère, J.C.: A new efficient algorithm for computing grobner bases (f4). Journal of Pure and Applied Algebra **139**(1-3) (1999) 61–68
12. Faugère, J.C., Gianni, P.M., Lazard, D., Mora, T.: Efficient computation of zero-dimensional gröbner bases by change of ordering. J. Symb. Comput. **16**(4) (1993) 329–344
13. Goubault-Larrecq, J., Roger, M., Verma, K.N.: Abstraction and resolution modulo ac: How to verify diffie-hellman-like protocols automatically. J. Log. Algebr. Program. **64**(2) (2005) 219–251
14. Lidl, R., Niederreiter, H.: Finite Fields (Encyclopedia of Mathematics and its Applications). Cambridge University Press (October 1996)
15. Peyrin, T.: Cryptanalysis of grindahl. In: ASIACRYPT. (2007) 551–567
16. Rijmen, V., Rompay, B.V., Preneel, B., Vandewalle, J.: Producing collisions for panama. In Matsui, M., ed.: FSE. Volume 2355 of Lecture Notes in Computer Science., Springer (2001) 37–51